1 Integer Linear Programs

An Integer Linear Program is a linear program where the variables are constrained to take integer values only. Note that all variables have to take only integer values.

An ILP can be written as

\[
\begin{align*}
\max & \quad c^T x \\
Ax & \leq b \\
x_i & \text{ is integral } \forall i.
\end{align*}
\]

The goal of this lecture is to present a technique for designing algorithms. We will give a template for designing combinatorial algorithms. The added advantage is that proving correctness comes for free. The template will come from linear programming; however the algorithms we design will be combinatorial.

To understand the template, we will try designing an iterative algorithm for the linear programming problem:

\[
\begin{align*}
\max & \quad c^T x \\
Ax & \leq b \\
(1)
\end{align*}
\]

The first step is to find some feasible solution. While this, in general, is as hard as solving an LP, in our examples, this will be easy. For instance, if \( b \geq 0 \), then \( x = 0 \) will work.

We will design an iterative step that will be repeatedly executed till we reach an optimum. Suppose we have a feasible \( x_i \) after the \( i \)th step. In the \( i + 1 \)th step, we will find a \( x_i + 1 \) which is also feasible, with increased cost.

At the beginning of the \( i + 1 \)th iteration we have an \( x_i \) such that \( Ax_i \leq b \). We need an \( x_i + 1 \) such that \( Ax_i + 1 \leq b \) and \( c^T (x_i + 1 - x_i) > 0 \).

In other words, we need to figure out a direction vector \( y_i \) which is \( x_i + 1 - x_i \). The cost must increase in this direction and moving in this direction should not land us outside the feasible region. How do we handle the second condition? It is clear that the inequalities which are equalities at \( x_i \) will play a role. For the strict inequalities, moving around \( x_i + 1 \) in a small enough neighbourhood will not lead to violations.

Let \( A'x' = b' \); \( A''x' < b'' \). For feasibility we need a \( y_i \) such that \( A'(x_i + \epsilon y_i) \leq b' \) for some small \( \epsilon > 0 \). Simplifying, we see that we need a \( y_i \) such that \( c^T y_i > 0 \) and \( A'y_i \leq 0 \).

Our objective now is: Design an algorithm to find such a \( y_i \) and update \( x_i \) suitably. Once we do this, we put this piece of code in a loop and we are done. This piece of code will be combinatorial and will depend on the problem at hand. Notice that this may be easier since the RHS is zero.

Exercise: Given such a \( y_i \), how do you find \( x_i + 1 \)?

Exercise: Prove that if \( x_i \) is not optimal, such a \( y_i \) exists.

Hint: Assume that \( x_0 \) is the optimal point. Now consider the vector \( \epsilon(x_0 - x^i) \). Note that this exercise provides a proof of correctness.

The final template is given below.
1. Find an initial feasible point $x^0$.

2. Loop as long as you can.
   
   In this loop we update $x^i$ to $x^{i+1}$ of increased cost. If we cannot, then the algorithm terminates.

Identify the equalities. That is, determine $A'$ such that $A'x^i = b'$; $A''x^i < b''$.

Find $y_i$ satisfying

$$c^T y_i = 1; A'y_i \leq 0$$

The previous step will be replaced by a combinatorial algorithm to compute $y_i$.

Determine the maximum $\epsilon > 0$ such that $x^i + \epsilon y_i$ is feasible. How?

$$x^{i+1} = x^i + \epsilon y_i.$$